IN THE CLAIMS:

1. (Currently Amended) A process for invoking a method of a server object in a distributed application in a distributed data processing system, the process comprising the computer-implemented steps of:

executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

executing said client object which begins an attempt to invoke a method in the server object;

in response to the client object beginning the attempt to invoke the method in the server object:

obtaining an object reference on the client for said second programming environment;

wrapping the object reference in an adapter which isolates the object reference from said client object, and the adapter performing data conversion between the object reference and the client object for different types of data[[;]]

~~calling by the client object the method in the server object, wherein said step of calling invokes a method of the adapter, the method invoked in the adapter corresponding to the method called in the server object;~~

~~calling, by the method invoked in the adapter, a method in the object reference that corresponds to the method invoked in the adapter; and~~

~~calling, by the object reference, the method in the server object, wherein the adapter causes the object reference to invoke the method in the server object transparently to the client object such that the client object is unaware of the operation of the object reference~~, wherein the step of performing data conversion between the object reference and the client object for different types of data comprises:

responsive to receiving a method called by the client object, parsing the method for an argument;

performing data conversion on the argument if necessary;

determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument;

delegating the method called by the client object to the object reference of the argument;

responsive to detecting a return value, determining if the return value is an object reference for the second programming environment;

if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and

returning the wrapped object reference to the client object.

2. (Original) The process of claim 1 wherein the adapter uses the object reference to invoke a method of a skeleton on the server.

3. (Original) The process of claim 1 wherein the skeleton invokes a method of a server object.

4. (Currently Amended) A method for implementing a distributed application in a distributed data processing system, the method comprising the computer-implemented steps of:

executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

in response to the client object beginning an attempt to invoke a method in the server object:

obtaining ~~an object reference~~ a proxy for the second programming environment; and

wrapping the proxy in an adapter which isolates the proxy from the client object, and the adapter performing data conversion between the proxy and the client object for different types of data[[; and]]

~~invoking a method of the adapter, wherein the adapter calls a corresponding method in the proxy~~, wherein the step of performing data conversion between the proxy and the client object for different types of data comprises:

responsive to receiving a method called by the client object, parsing the method for an argument;

performing data conversion on the argument if necessary;

determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve a proxy of the argument, wherein the proxy of the argument provides a reference to a server object in the second programming environment representing the argument;

delegating the method called by the client object to the proxy of the argument;

responsive to detecting a return value, determining if the return value is a proxy for the second programming environment;

if the return value is proxy for the second programming environment, wrapping the proxy with a suitable adapter based on a type of the object reference; and

returning the wrapped proxy to the client object.

5. (Previously presented) The method of claim 4 wherein the adapter is a Java class that implements an interface supported by the server object.

6. (Previously presented) The method of claim 4 wherein the server object is an Enterprise JavaBean.

7. (Previously presented) The method of claim 4 wherein the object reference is obtained from a naming service.

8. (Previously presented) The method of claim 4 wherein the proxy is a Common Object Request Broker Architecture (CORBA) proxy.

9. (Original) The method of claim 8 wherein the adapter calls a method of the CORBA proxy.

10. (Original) The method of claim 8 wherein the CORBA proxy is a Java class residing on a client computer.

11. (Original) The method of claim 8 wherein the CORBA proxy passes the method request to an object request broker.

12. (Currently Amended) A data processing system for invoking a method of a server object in a distributed application in a distributed data processing system, the data processing system comprising:

execution means for executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

execution means for executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

execution means for executing said client object which begins an attempt to invoke a method in the server object;

in response to the client object beginning the attempt to invoke the method in the server object:

obtaining means for obtaining an object reference on the client for said second programming environment;

wrapping means for wrapping the object reference in an adapter which isolates the object reference from the client object, and the adapter performing data conversion between the object reference and the client object for different types of data[[;]]

~~calling means for calling by the client object the method in the server object which invokes a method of the adapter, the method invoked in the adapter corresponding to the method called in the server object;~~

~~calling means for calling, by the method invoked in the adapter, a method in the object reference that corresponds to the method invoked in the adapter; and~~

~~calling means for calling, by the object reference, the method in the server object, wherein the adapter causes the object reference to invoke the method in the server object transparently to the client object such that the client object is unaware of the operation of the object reference~~, wherein the means for performing data conversion between the object reference and the client object for different types of data comprises:

responsive to receiving a method called by the client object, parsing means for parsing the method for an argument;

performing means for performing data conversion on the argument if necessary;

determining means for determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, unwrapping means for unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument;

delegating means for delegating the method called by the client object to the object reference of the argument;

responsive to detecting a return value, determining means for determining if the return value is an object reference for the second programming environment;

if the return value is an object reference for the second programming environment, wrapping means for wrapping the object reference with a suitable adapter based on a type of the object reference; and

returning means for returning the wrapped object reference to the client object.

13. (Original) The data processing system of claim 12 wherein the adapter uses the object reference to invoke a method of a skeleton on the server.

14. (Original) The data processing system of claim 12 wherein the skeleton invokes a method of a server object.

15. (Currently Amended) A data processing system for implementing a distributed application in a distributed data processing system, the data processing system comprising:

execution means for executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

execution means for executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

in response to the client object beginning an attempt to invoke a method in the server object:

obtaining means for obtaining an object reference for the second programming environment;

wrapping means for wrapping the proxy in an adapter which isolates the proxy from the client object, and the adapter performing data conversion between the proxy and the client object for different types of data[[; and]]

~~invoking means for invoking a method of the adapter, wherein the adapter calls a corresponding method in the proxy~~, wherein the means for performing data conversion between the proxy and the client object for different types of data comprises:

responsive to receiving a method called by the client object, parsing means for parsing the method for an argument;

performing means for performing data conversion on the argument if necessary;

determining means for determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, unwrapping means for unwrapping the adapter from the argument to retrieve a proxy of the argument, wherein the proxy of the argument provides a reference to a server object in the second programming environment representing the argument;

delegating means for delegating the method called by the client object to the proxy of the argument;

responsive to detecting a return value, determining means for determining if the return value is a proxy for the second programming environment;

if the return value is proxy for the second programming environment, wrapping means for wrapping the proxy with a suitable adapter based on a type of the object reference; and

returning means for returning the wrapped proxy to the client object.

16. (Original) The data processing system of claim 15 wherein the adapter is a Java class that implements an interface supported by the server object.

17. (Original) The data processing system of claim 15 wherein the server object is an Enterprise JavaBean.

18. (Original) The data processing system of claim 15 wherein the object reference is obtained from a naming service.

19. (Original) The data processing system of claim 15 wherein the proxy is a Common Object Request Broker Architecture (CORBA) proxy.

20. (Original) The data processing system of claim 19 wherein the adapter calls a method of the CORBA proxy.

21. (Original) The data processing system of claim 19 wherein the CORBA proxy is a Java class residing on a client computer.

22. (Original) The data processing system of claim 19 wherein the CORBA proxy passes the method request to an object request broker.

23. (Currently Amended) A computer program product in a computer readable medium for use in a data processing system for invoking a method of a server object in a distributed application in the distributed data processing system, the computer program product comprising:

instructions for executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

instructions for executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

instructions for executing said client object which begins an attempt to invoke a method in the server object;

in response to the client object beginning the attempt to invoke the method in the server object:

instructions for obtaining an object reference on the client for said second programming environment;

instructions for wrapping the object reference in an adapter which isolates the object reference from the client object, and the adapter performing data conversion between the object reference and the client object for different types of data[[;]]

~~instructions for calling by the client object the method in the server object which invokes a method of the adapter, the method invoked in the adapter corresponding to the method called in the server object;~~

~~instructions for calling, by the method invoked in the adapter, a method in the object reference that corresponds to the method invoked in the adapter; and~~

~~instructions for calling, by the object reference, the method in the server object, wherein the adapter causes the object reference to invoke the method in the server object transparently to the client object such that the client object is unaware of the operation of the object reference~~, wherein the instructions for performing data conversion between the object reference and the client object for different types of data comprises:

responsive to receiving a method called by the client object, instructions for parsing the method for an argument;

instructions for performing data conversion on the argument if necessary;

instructions for determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, instructions for unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument;

instructions for delegating the method called by the client object to the object reference of the argument;

responsive to detecting a return value, instructions for determining if the return value is an object reference for the second programming environment;

if the return value is an object reference for the second programming environment, instructions for wrapping the object reference with a suitable adapter based on a type of the object reference; and

instructions for returning the wrapped object reference to the client object.

24. (Currently Amended) A computer program product in a computer readable medium for use in a data processing system for implementing a distributed application in a distributed data processing system, the computer program product comprising:

instructions for executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;

instructions for executing a server object in a server that implements a second programming environment that is different from said first programming environment, said server object being written for said first programming environment;

in response to the client object beginning an attempt to invoke a method in the server object:

instructions for obtaining an object reference for the second programming environment;

instructions for wrapping the proxy in an adapter which isolates the proxy from the client object, and the adapter performing data conversion between the proxy and the client object for different types of data[[; and]]

~~instructions for invoking a method of the adapter, wherein the adapter calls a corresponding method in the proxy~~, wherein the instructions for performing data conversion between the proxy and the client object for different types of data comprises:

responsive to receiving a method called by the client object, instructions for parsing the method for an argument;

instructions for performing data conversion on the argument if necessary;

instructions for determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, instructions for unwrapping the adapter from the argument to retrieve a proxy of the argument, wherein the proxy of the argument provides a reference to a server object in the second programming environment representing the argument;

instructions for delegating the method called by the client object to the proxy of the argument;

responsive to detecting a return value, instructions for determining if the return value is a proxy for the second programming environment;

if the return value is proxy for the second programming environment, instructions for wrapping the proxy with a suitable adapter based on a type of the object reference; and

instructions for returning the wrapped proxy to the client object.